

AF  
I-24



**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

re Application of:

Saulpaugh, et al.

Serial No. 09/693,673

Filed: October 19, 2000

For: Message Gates Using a  
Shared Transport in a  
Distributed Computing  
Environment

§ Group Art Unit: 2144  
§  
§ Examiner: Maniwang, Joseph R.  
§  
§ Atty. Dkt. No.: 5181-64200  
§ P4998

<p style="text-align: center;">CERTIFICATE OF MAILING 37 C.F.R. § 1.8</p> <p>I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:</p> <p style="text-align: center;"><u>Robert C. Kowert</u> Name of Registered Representative</p> <p><u>December 10, 2004</u>      <u>[Signature]</u> Date                                      Signature</p>
---

**APPEAL BRIEF**

**Mail Stop Appeal Brief - Patents**  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed October 15, 2004, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

12/14/2004 EABUBAK1 00000025 501505 09693673

01 FC:1402 500.00 DA

## **I. REAL PARTY IN INTEREST**

The subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054, as evidenced by the assignment recorded at Reel 011250, Frame 0109.

## **II. RELATED APPEALS AND INTERFERENCES**

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

## **III. STATUS OF CLAIMS**

Claims 1-24 are pending and rejected. The rejection of claims 1-24 is being appealed. A copy of claims 1-24 is included in the Claims Appendix attached hereto.

## **IV. STATUS OF AMENDMENTS**

No amendments to the claims have been submitted subsequent to the final rejection.

## **V. SUMMARY OF CLAIMED SUBJECT MATTER**

Claim 1 is directed to a device in a distributed computing environment that includes one or more clients and a plurality of message gates, wherein each message gate is configured for sending and receiving messages for one of said clients in a data representation language to and from a respective paired message gate at another device in the distributed computing environment. A message gate is the message endpoint for a client or service in a distributed computing environment. For example, Figure 10 illustrates how both client 110 and service 112 may each construct a message gate 130 for

communicating. (See also, Figures 8, 11a, 11b, 12, page 13, lines 4-14, page 28, lines 13-22, and page 30, line 27 – page 31, line 7). A message gate may provide a secure message endpoint that sends and receives type-safe messages (see page 31, lines 9-14). The messages may be in a data representation language such as eXtensible Mark-up Language (XML). (See, Figure 6, 11b, 13, 34, 35, page 32, line 28- page 33, line 8, page 22, lines 10-19). For a client, a message gate may represent the authority to use some or all of a service's capabilities. Each capability may be expressed in terms of a message that may be sent to a service. (See, page 29, line 26 – page 30, line 7, page 86, lines 26 – 28, page 89, lines 4-17).

The device also includes a message transport configured to implement a transport protocol for sending and receiving messages. For instance, message gates may allow clients and services to exchange data representation language messages in a secure and reliable fashion over any suitable message transport (e.g. HTTP). (See Figure 3, 4, 5, 7, 10a, 12, 24, 48, page 23, lines 1- page 24, line 30, page 46, line 12-19, page 146, lines 13-21).

Furthermore, each of the message gates may reference the message transport, wherein each message gate is configured to send and receive messages independently of other message gates while sharing the message transport for implementing the transport protocol for sending and receiving messages. For instance, a gate may be implemented as a layer above a device's transport layer (e.g. networking sockets). Each gate may include a transport reference. The gate name may be bound to the transport reference. Multiple gates may share the same message transport. For example, Figure 48 illustrates multiple clients and gates sharing the same transport layer 106. (See also, figure 3, 5, 10a, and 12). In one embodiment, multiple gates may include transport references to the same TCP/IP socket. By sharing the same message transport, the size and complexity of each gate may be reduced. In some embodiments, the transport reference may be a transport URI (e.g. URL) or socket reference and may provide a mechanism for naming an underlying transport and sharing the transport with other gates. As described in the

Specification on pages 46-47, multiple local gates may include a reference to the same transport, however, each local gate may behave independently of the other local gates sending and receiving messages to and from its paired remote gate. Each local gate may behave independently of the other local gates in that each local gate need not be aware of or communicate with the other local gates. Each gate may use the transport reference to access the same transport layer without knowledge of the other gates' activities.

Independent claim 9 is directed to a method for sending and receiving messages in a distributed computing environment. The method recited in claim 9 includes a first message gate referencing a message transport to send first messages to a first destination address, wherein said first messages are formatted in a data representation language and a second message gate referencing the message transport to send second messages to a second destination address, wherein said second messages are formatted in a data representation language. Thus, the first and second message gates may share a single message transport to send messages in a data representation language (such as XML). The sharing of a message transport by multiple message gates is described above. The method of claim 9 further includes the message transport implementing a transport protocol for sending the first and second messages to the first and second destination addresses respectively, as described above regarding claim 1. Additionally, the message gates independently share the message transport for sending messages to the destination addresses. Please refer to the description above for more information regarding message gates independently sharing a message transport.

Independent claim 17 is directed to a carrier medium comprising program instructions that are executable to implement functionality similar to that recited by independent claim 9, described above.

## **VI. GROUND OF REJECTION TO BE REVIEWED ON APPEAL**

1. Claims 1, 9, 16, 17 and 24 are rejected under 35 U.S.C. § 102(b) as being anticipated by Hill et al. (U.S. Patent 5,511,197).

2. Claims 1-4, 8-12, 15, 17-20 and 23 are rejected under 35 U.S.C. § 102(b) as being anticipated by Serlet et al. (U.S. Patent 5,481,721).

3. Claims 1-4, 8-12, 15, 17-20 and 23 are rejected under 35 U.S.C. § 102(e) as being anticipated by Marcos et al. (U.S. Patent 6,347,342).

4. Claims 1, 6-9, 14-17 and 22-24 are rejected under 35 U.S.C. § 102(b) as being anticipated by Kingdon (U.S. Patent 5,349,642).

5. Claims 5, 13 and 21 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Marcos et al. (U.S. Patent 6,347,342) in view of Bergman et al. (U.S. Patent 6,564,263).

## **VII. ARGUMENT**

### **First Ground of Rejection:**

Claims 1, 9, 16, 17 and 24 stand finally rejected under 35 U.S.C. § 102(b) as being anticipated by Hill et al. (U.S. Patent 5,511,197 – hereinafter, Hill). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

#### **Claims 1, 9 and 17:**

Hill does not teach a plurality of message gates each configured for sending and receiving messages for one of the clients in a data representation language to and from a respective paired message gate at another device in the distributed computing environment, as recited in claim 1. Regarding claim 1, the Examiner, citing column 7, lines 19-42 of Hill, states, “[t]he stub and proxy objects sent and received messages, acting as message gates as claimed.” However, Hill fails to disclose sending and receiving messages *in a data representation language*. Instead, Hill discloses stub and

proxy objects that implement remote procedure calls and communicate through the marshalling and unmarshalling of pointers, method names, and parameters into messages by inserting their *actual values* in the message (Hill, column 1, lines 25-45; column 7, lines 19-25; column 8 lines 20-23).

In response to Appellants' previous arguments, the Examiner argues that the "broad concept of communicating messages in a data representation language as claimed relates to nothing more than a standard method or format of the messages communicated" and that "Hill disclosed such a limitation, stating that stub channels utilized a certain protocol to communicate, thus disclosing communication of messages in a data representation language." (Final Office Action dated May 14, 2004, Response to Arguments, paragraph 24). The Examiner has incorrectly interpreted Hills remote procedure calls as sending messages in a data representation language. **Sending messages in a data representation language is not merely the use of any standard communication method or format, and is distinctly different from sending messages using, for example, stubs and proxies, RMI or another code based communication format, as used by Hill.** Data representation languages are known in the art as languages that are used to represent or describe data in documents. The eXtensible Markup Language is one example of such a data representation language. Data representation languages are not traditionally used in the prior art for programmatic interfaces such as the stub and proxy communications in Hill. Thus, Hill clearly does not anticipate sending and receiving messages in a data representation language.

In contrast, Hill describes a remote procedure call (RPC) mechanism. The RPC mechanisms described in Hill *by definition* do not employ data representation language messages. A data representation language is a specific type of language having a specific purpose for describing data documents, as is well known to those of ordinary skill in the art. In the prior art, data representation languages have been used to describe data documents or content (such as HTML or XML documents that may be displayed or processed in Internet communications). In Hill, clients use a programmatic (code-based)

messaging RPC interface to invoke services. Such programmatic (code-based) interfaces are also discussed in the Related Art section of the present application and *by definition* do not employ data representation language messages. The messages in Hill are clearly not data representation language messages.

The Examiner argues that since Hill teaches the use of a “certain protocol to communicate” (citing column 20, lines 29-31), Hill discloses sending and receiving messages in a data representation language. Following the Examiner’s line of reasoning, the use of any protocol teaches the sending and receiving of messages in a data representation language. Such an interpretation is clearly incorrect. The sending and receiving of messages in a data representation language to and from respective paired message gates requires more than a particular protocol.

The Examiner further argues in the Advisory Action of October 6, 2004 that a second reference (RFC 1014 – XDR: External Data Representation Standard) discloses the use of a data representation language in RPC protocols. However, the Examiner’s reliance upon RFC 1014 as a second reference is clearly improper in a § 102(b) anticipation rejection. “A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently, in a single prior art reference.” *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987) (emphasis added). There is no evidence that the RPC mechanism of Hill would inherently use the XDR standard described in the additional reference. Thus, the Examiner’s reliance on the RFC 1014 reference is improper.

Additionally, even if the RFC 1014 reference was considered in combination with Hill, such a combination would still not teach paired message gates configured for sending and receiving of messages in a data representation language. RFC 1014 only teaches that “[p]rotocols such as Sun RPC (Remote Procedure Call) and the NFS (Network File System) use XDR *to describe the format of their data*” (emphasis added). Thus, RFC 1014 does not disclose anything about sending and receiving messages in

XDR or any data representation language. Instead, RFC 1014 only mentions using XDR in a traditional manner to describe the format of data. RFC 1014 does not state that RPC protocols send messages in a data representation language. Furthermore, since neither Hill nor any other art of record mentions anything about using XDR or any other data representation language for sending messages between paired message gates, there is no suggestion to modify Hill's system to send messages in a data representation language.

### **Claims 16 and 24:**

Claims 16 and 24 are patentable for at least the reasons given above regarding their respective independent claims. Additionally, Hill does not teach binding a first gate name for the first message gate to a transport reference for the message transport, wherein the first message gate is configured to receive messages at an address including a combination of the first gate name and the transport reference, as recited in Appellants' claim 16. The Examiner cites column 10, lines 30-32, where Hill describes including an inter-process communication message address for a stub object in a message between a server and a client. However, Hill does not describe the message address as gate name bound to a transport reference, nor as a combination of a gate name and a transport reference. Hill does not describe his stub object receiving messages at an address that is a *combination of the message address and a transport reference*. Instead, Hill only describes how a message address for stub object 302 is sent to a client and how the client uses the received message address when instantiating proxy object 303 (Hill, column 7, lines 2-18). Thus, Hill clearly does not teach a message gate configured to receive messages at an address including a combination of a gate name and a transport reference. The Examiner is merely assuming in hindsight that Hill's system includes binding a gate name to a transport reference without providing any argument or reasoning to support that assumption. The Examiner has not cited any portion of Hill that discloses any message gate configured to receive messages at an address including a combination of a gate name and a transport reference.

### **Second Ground of Rejection:**



Claims 1-4, 8-12, 15, 17-20 and 23 stand finally rejected under 35 U.S.C. § 102(b) as being anticipated by Serlet et al. (U.S. Patent 5,481,721 – hereinafter Serlet). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

**Claims 1, 8, 9 15, 17 and 23:**

Serlet does not teach a plurality of message gates each configured for sending and receiving messages for one of the clients in a data representation language to and from a respective paired message gate at another device in the distributed computing environment, as recited in claim 1. The Examiner states that Serlet discloses a “proxy object [that] provided a way to send and receive messages on behalf of objects, thus providing message gates as claimed.” However, Serlet discloses proxies that communicate through Mach messages and that a Mach message contains “a header followed by zero or more data objects” (Serlet, column 9, lines 20-25). Additionally, Serlet describes how, for efficiency, “large array arguments are passed out-of-line with copy-on-write semantics” (Serlet, column 9, lines 5-9). Serlet further teaches that “[t]he argument encoding for standard C data types is explicit and strictly pass-by-value, and maps *substantially directly* onto a Mach message” (Serlet, column 12, lines 46-49). Since, Serlet’s Mach messages are clearly not in a data representation language, there is no disclosure in Serlet of a system including message gates configured for sending and receiving messages in a data representation language.

In response to Appellants’ previous arguments, the Examiner maintains that “a data representation language as claimed is broadly interpreted as a standard format or method for communicating data” and that “Serlet disclosed such a limitation, stating that messages sent between a sender and receiver must be encoded in a form understandable by both sides, providing a type of protocol for communicating data.” (Final Office Action dated May 14, 2004, Response to arguments, paragraph 25, and Advisory Action dated October 14, 2004). Such an interpretation is clearly incorrect. Sending messages in

a data representation language requires more than just using any standard method or format. Claim 1 does not state that messages can be in any standard format. Instead, claim 1 recites messages in a data representation language. A data representation language is a specific type of language traditionally used for describing documents, not messages for a programmatic interface as in Serlet. Thus, there is no basis for the Examiner's assumption that Serlet's system includes the use of messages in a data representation language. Data representation language messages are distinctly different from Serlet's Mach messages or messages sent using other code-based communication formats. Serlet clearly does not *anticipate* sending and receiving messages in a data representation language.

In the Advisory Action, the Examiner also incorrectly characterizes Appellants' previous argument. Appellants have argued previously that Serlet's system does not include the use of messages in a data representation language because a data representation language is a specific type of language traditionally used for describing documents, and not used for messages in a programmatic interface, as in Serlet. The Examiner replies, "it is noted that features upon which applicant relies (i.e. the use of data representation languages to describe documents) are not recited in the rejected claim(s)." However, Appellants never argued that claim 1 recited such a limitation. Instead, Appellants have simply pointed out that data representation languages have traditionally been used to describe data in documents and that Serlet uses code-centric, RPC type protocols which do not use data representation language messages. Thus, without some clear teaching that Serlet's system includes sending and receiving messages in a data representation language, Serlet cannot be said to anticipate Appellants' claimed invention.

Additionally, in the Advisory Action, the Examiner refers to RFC 1014 to argue that remote procedure call (RPC) protocols utilized data representation languages. As noted above, the Examiner's reliance on the additional RFC 1014 is improper for a rejection based on the anticipation standard. There is no evidence that Serlet would

inherently use the XDR standard described in the additional RFC 1014 reference. Moreover, as noted above for first ground of rejection, RFC 1014 does not teach that RPC protocols send and received message in a data representation language, but merely that such protocols may use XDR “to describe the format of their data” (RFC 1014, section 1, paragraph 2). Thus, even if the RFC 1014 reference was considered in combination with Serlet, such a combination would still not teach paired message gates configured for sending and receiving of messages in a data representation language. Furthermore, since neither Serlet nor any other art of record mentions anything about using XDR or any other data representation language for sending messages between paired message gates, there is no suggestion to modify Serlet’s system to send messages in a data representation language.

**Claims 2, 10 and 18:**

Serlet does not teach wherein each message gate is configured to verify messages according to a data representation language message schema, as recited by claim 2. The Examiner cites column 12, lines 30-45 of Serlet. However, the cited passage of Serlet describes dynamically determining a communication protocol by requesting “the signature corresponding to the actual class implementation that will ultimately receive the message.” A signature corresponding to a class implementation cannot be considered a data representation language message schema. Serlet describes how such a method signature is “a domain-independent encapsulation of the method name, its argument types and its return value type” (Serlet, column 12, lines 26-29). Serlet does not mention any sort of data representation language message schema, nor does Serlet mention anything about verifying messages according to a data representation language message schema. Serlet does not describe the class implementation signature as being implemented in or communicated as a data representation language message schema.

Furthermore, Serlet fails to teach *verifying* messages at all. Serlet teaches using the received signature to generate appropriate argument types for a message. However, *generating* messages using a class implementation signature is not the same as *verifying*

messages according to a *data representation language message schema*.

**Claims 3, 11 and 19:**

Serlet fails to teach wherein each message gate is configured to verify type correctness of messages sent or received through that message gate according to the data representation language message schema. The Examiner states only that “Serlet disclosed ver[if]ying messages for type and form of encoding” and cites column 12, lines 30-45 of Serlet. However, as mentioned above regarding the rejection of claim 2, Serlet teaches only the use of a class implementation signature to dynamically determine a protocol and to determine how to encode arguments in messages. Serlet does not teach that such a signature is used to *verify* the type correctness of messages. Serlet does not teach that a class implementation signature is a data representation language message scheme. Verifying type correctness according to a data representation language schema is clearly distinct and different from generating messages by encoding arguments according to a received class implementation signature. Additionally, as mentioned above regarding the rejection of claim 2, Serlet fails to mention anything about verifying messages.

**Claims 4, 12 and 20:**

Serlet does not anticipate wherein each message gate is configured to verify format correctness of messages sent or received through that message gate according to the data representation language message schema. The Examiner cites only column 12, lines 30-45 where Serlet describes requesting a class implementation signature to dynamically learn “how to encode arguments for each message as [that message] is encountered.” As mentioned above regarding the rejections of claims 2 and 3, Serlet fails to teach that such a signature has anything to do with a data representation language schema. Additionally, Serlet does not mention that such a class implementation signature can be used to verify format correctness of messages sent or received through a message gate. Thus, not only does Serlet fail to teach the use of a data representation language schema, Serlet also does not disclose verifying format correctness of messages. Even if

Serlet's signature could be considered a schema (which Appellants maintain it cannot). Serlet still does not teach using the signature to verify format correctness. Instead, Serlet only describes using such a signature to learn how to encode the arguments for a message. Serlet does not teach that the class implementation signature describes the format of message nor does Serlet teach that such a signature can be used to verify format correctness of messages.

Furthermore, Serlet clearly teaches the use of a Mach message that contains a header followed by zero or more data objects. Serlet also describes the method signature (which the Examiner is equating to a data representation language message schema) as including only the method name, its argument types, and its return type. Thus, even if Serlet were to teach verify messages, which he clearly doesn't, the signature information cited by the Examiner could not be used to verify format correctness of Serlet's messages.

### **Third Ground of Rejection:**

Claims 1-4, 8-12, 15, 17-20 and 23 stand finally rejected under 35 U.S.C. § 102(e) as being anticipated by Marcos et al. (U.S. Patent 6,347,342 – hereinafter Marcos). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

### **Claims 1, 8, 9, 15, 17 and 23:**

Marcos does not disclose a system including a plurality of message gates wherein each message gate is configured for sending and receiving messages in a data representation language. Regarding claim 1, the Examiner states that Marcos teaches that a "connection between client objects and server objects was created through proxy and stub objects, which allowed for sending and receiving messages, thus providing message gates as claimed." Marcos teaches using a distributed object model or protocol to forward messages (Marcos, column 6, line 66 to column 7, line 3). Marcos discloses using code that will "encode and decode an operation and its parameters into a compacted

message format” (Marcos, column 2, lines 12-14) which the mediating component translates or maps from one object model to another (Marcos, column 7, lines 16-24). Thus, Marcos does not disclose that each message gate is configured for sending and receiving messages for one of said clients in a data representation language.

In response to Appellants’ previous arguments, the Examiner contends that “Marcos disclosed the use of a protocol for communicating messages between client objects and server objects, thus disclosing message gates configured to communicate messages in a data representation language.” (Final Office Action dated May 14, 2004, Response to arguments, paragraph 26). The Examiner maintains that sending messages in any format implies sending messages is a data representation language. However, sending messages in a data representation language requires more than just a standard method or format, and is distinctly different from sending messages using OLE/COM, CORBA, or another code based communication format as in Marcos. By definition, OLE/COM, CORBA, and other code based communication formats as in Marcos do not use data representation language messages. Thus, Marcos does not anticipate sending and receiving messages in a data representation language.

In response to Appellants previous arguments, the Examiner argues that Marcos use of a “compacted message format” describes sending messages in a data representation language. However, a “compacted message format” does not imply a data representation language. Data representation languages are not considered *compacted* message formats.

Further, Appellants note that the Examiner’s argument amounts to stating that sending messages in a data representation language is inherent in Marcos’ teaching. Appellants respectfully remind the Examiner that for a prior art reference to be shown to inherently teach something, the Examiner must show evidence that makes it “clear that the missing descriptive matter is *necessarily* present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill.” (emphasis added) (M.P.E.P. § 2131.01, paragraph 4). In response, the Examiner, in the Advisory

Action, states, “the Examiner can find no such instance in previous arguments or rejections.” The issue is not whether or not the Examiner explicitly states that he is arguing that the use of data representation language messages is inherent in Marcos. The Examiner’s contention that Marcos teaches something (such as sending data representation language messages) that is not mentioned anywhere by Marcos clearly amounts to a theory of inherency. This is further supported by the fact that the Examiner is attempting to use the RFC 1014 reference to insert the use of data representation language messages into Marcos’ teachings. However, data representation languages are not inherent in the protocol for communicating messages between client and server objects using NEXT’s DO, COM, OLE, or CORBA, as disclosed by Marcos. In fact, it is well understood in the art that code-centric mechanism such as NEXT’s DO, COM, OLE, or CORBA specifically do not use data representation language messages.

In the Advisory Action, the Examiner refers to RFC 1014 to argue that remote procedure call (RPC) protocols utilized data representation languages. As noted above, the Examiner’s reliance on the additional RFC 1014 is improper for a rejection based on the anticipation standard. There is no evidence that Marcos would inherently use the XDR standard described in the additional RFC 1014 reference. Moreover, as noted above for first ground of rejection, RFC 1014 does not teach that RPC protocols send and received message in a data representation language, but merely that such protocols may use XDR “to describe the format of their data” (RFC 1014, section 1, paragraph 2). Thus, even if the RFC 1014 reference was considered in combination with Marcos, such a combination would still not teach paired message gates configured for sending and receiving of messages in a data representation language. Furthermore, since neither Marcos nor any other art of record mentions anything about using XDR or any other data representation language for sending messages between paired message gates, there is no suggestion to modify Marcos’s system to send messages in a data representation language.

**Claims 2, 10 and 18:**

Claims 2, 10, and 18 are patentable for at least the reasons given above regarding their respective independent claims. Additionally, regarding claim 2, the Examiner states, “Marcos disclosed proxy and stub objects to have the ability to verify object type.” Appellants disagree with the Examiner’s interpretation of Marcos. Marcos fails to teach verifying messages according to a data representation language *message schema*. Marcos discloses using messages comprising methods and method arguments (Marcos, column 6, lines 62-63; column 4, lines 23-25; column 15, lines 27-28; column 15, lines 64 – column 16, lines 4). Marcos also teaches translating arguments from one object model to another (Marcos, column 16, lines 42-50). Marcos makes no reference to any data representation language message schema. According to Marcos, the type of an argument from one object model is compared with an expected type in another object model, but Marcos fails to teach verifying *messages according to a data representation language message schema*.

In response to Appellants’ previous arguments, the Examiner argues, “in its broadest sense, a data representation language message schema relates to nothing more than a set format for communicating data that messages must adhere to in order for a process to interpret them” and that “Marcus taught the use of such a format, stating that objects could communicate using specific data types, thus providing a message schema.” The Examiner further contends that “verification of these data types was implicitly disclosed, as Marcos disclosed the possibility of translating data types into a type compatible with a receiving object, the need for such translation clearly determined only through a verification of the message data type according to a given schema acceptable by the receiving object.” (Final Office Action dated May 14, 2004, Response to arguments, paragraph 27). However, verifying messages according to a data representation language message schema is very different than simply teaching that objects may be communicated using specific data types and also very different from translating data types as taught by Marcos. Specifically, a message containing data in a specific data type can be translated to a different data types without verifying that the messages conforms to a particular schema, such as according to a data representation language schema. In other words,



message content can be translated between data types based purely on a mapping of data types and does not imply verifying the message according to a data representation language message schema. This is in fact the type of translation Marcos teaches. For instance, Marco states, “[w]hen a message is directed to the server object via the proxy object, mediating component 204 performs a mapping, or translation *of arguments* for use by the server object” (emphasis added, Marcos, column 7, lines 16-19).

Additionally, Marcos teaches, “when a connection request is received ... mediating component 204 determines whether a server object ... can service the message” and continues to state, “mediating component 204 queries information available in object model B to determine whether a service object exists in that system and created in that model that can process the message using the arguments supplied by the client object.” (Marcos, column 7, lines 6- 12). Thus, Marcos actually teaches that, rather than verify a message according to a data representation language schema, a mediating component searches for a server object that can process the message as is. These are two very different concepts. Thus, Marcos clearly does not anticipate verifying messages according to a data representation language message schema.

**Claims 3, 11 and 19:**

Marcos fails to teach wherein each message gate is configured to verify type correctness of messages sent or received through that message gate according to the data representation language message schema, as recited in claim 3. The Examiner argues that Marcos “disclosed proxy and stub objects to have the ability to verify object type and cites column 15, line 60 through column 16, line 55. However, the cited passage only refers the use of a mediating component (such as a proxy or stub object) that compares the sent argument type with an expected argument type. Marcos does not describe the argument types as being specified in a data representation language message schema. Instead, Marcos teaches the translation of arguments from one object model to another (Marcos, column 15, line 64-column 16, line 5). Marcos states that arguments that “are unique to, or incompatible with, the argument types of another environment” are

translated (Marcos, column 16, lines 16-32). Thus, Marcos is not teaching the verification of type correctness of messages, but rather Marcos assumes that the argument types in a message are correct and translates them to different types if needed. In other words, if an argument type in a message does not match the type expected by the receiving object model, Marcos assumes the type is correct for the sending object model and therefore translates it for use by the receiving object model.

Furthermore, Marcos translation process cannot be considered verifying type correctness of messages according to a data representation language message schema. Marcos never mentions a data representation language message schema. The Examiner argues, referring to claim 2 in the Advisory Action, that “a data representation language message schema amounts to nothing more a set format for communicating data that messages must adhere to” and that “Marcos taught the use of such a format.” (The Examiner is presumably relying upon a similar interpretation in his rejection of claim 3 since the Examiner has provided only a single argument for rejections of both claims 2 and 3.) However, just using a set format for communicating data does not imply the verification of messages according to a data representation language message schema.

Additionally, as the Examiner admits, Marcos teaches both the translation of argument types and the searching for a specific server object to handle a received message. Thus, Marcos does not teach a set format that messages much adhere to, as the Examiner suggests. The Examiner’s interpretation of Marcos’ system does not follow the Examiner’s own line of reasoning. Thus, not only does Marcos fail to anticipate message gates configured to verify type correctness of messages according to the data representation language message schema, the Examiner argument does not even apply to his own interpretation of Marcos.

Furthermore, claims 3, 11 and 19 are patentable for at least the reasons given above regarding their respective independent claims.

**Claims 4, 12 and 20:**

Marcos fails to disclose wherein each message gate is configured to verify format correctness of messages sent or received through that message gate according to the data representation language message schema, as recited in claim 4. The Examiner cites column 15, line 60 through column 16, line 55 of Marcos. However, this passage only refers to the use of proxy or stub objects as mediating components that compare sent argument types with expected argument types and perform translations if necessary. Nowhere does Marcos mention verifying format correctness of message according to a data representation language message schema. Translating argument types in messages is very different from verifying format correctness of messages.

Marcos does not describe argument types as being verifiable according a data representation language message schema. Instead, Marcos teaches the translation of arguments from one object model to another (Marcos, column 15, line 64-column 16, line 5). Marcos states that arguments that “are unique to, or incompatible with, the argument types of another environment” are translated (Marcos, column 16, lines 16-32). Thus, Marcos is not teaching the verification of format correctness of messages, but rather Marcos only teaches the translation of argument types. The Examiner has not pointed out any portion of Marcos that describes verifying format correctness of messages. Even if Marcos’ argument types could be considered a data representation language message schema (which they clearly are not) Marcos still does not teach using the argument types to verify format correctness or messages. Instead, Marcos only describes translating argument types if necessary. Thus, Marcos fails to anticipate messages gates configured to verify format correctness of messages according to a data representation language message schema. Furthermore, claims 4, 12, and 20 are patentable for at least the reasons given above regarding their respective independent claims.

**Fourth Ground of Rejection:**

Claims 1, 6-9, 14-17 and 22-24 stand finally rejected under 35 U.S.C. § 102(b) as

being anticipated by Kingdon et al (U.S. Patent 5,349,642 – hereinafter Kingdon). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

**Claims 1, 6, 7, 8, 9, 14, 15, 17, 22 and 23:**

Kingdon does not teach a system wherein each message gate is configured for sending and receiving messages for one of the clients in a data representation language. Regarding claim 1, the Examiner states, “Kingdon disclosed ... a client and server [that] make use of client and server stubs connected to a transport for communicating with each other, thus providing paired message gates as claimed.” Kingdon teaches the use of a very specific message packet format consisting of a length header, a request code representing the particular type of procedure being requested by the client, and data (Kingdon, column 5, lines 27-36, and Figure 3A). Kingdon’s message packet format is very different from a message in a data representation language. Kingdon clearly fails to teach the use of a data representation language. In response to Appellants’ previous arguments, the Examiner argues, “a data representation language as claimed is broadly interpreted as a standard format or method for communicating data” and further argues that “Kingdon thus disclosed communicating messages in a data representation language, as messages were communicated in packets created with a specific format.” (Final Office Action dated May 14, 2004, Response to arguments, paragraph 28). Appellants respectfully disagree and argue that sending messages in a data representation language is not just a standard method or format, but is distinctly different from sending messages using Kingdon’s unique message packet format. Please see the previous arguments regarding the Examiner’s other grounds of rejections for claim 1 for more arguments regarding the differences between messages in a data representation language and other standard formatting of messages. Kingdon clearly does not anticipate sending and receiving messages in a data representation language.

For the same reasons given in regard to the other grounds of rejection, Appellants assert that the Examiner’s reliance on the RFC 1014 reference in the Advisory Action is

improper.

**Claims 16 and 24:**

Kingdon does not teach binding a first gate name for the first message gate to a transport reference for the message transport, wherein the first message gate is configured to receive messages at an address including a combination of the first gate name and the transport reference, as recited in Appellants' claim 16. The Examiner cites column 2, lines 50-52 and 60-52, where Kingdon describes including source and destination addresses in message packets. However, Kingdon does not describe these source and destination addresses as message gate names that are bound to transport references. Nor does Kingdon describe them as being addresses including a combination of a gate name and a transport reference. Kingdon is clearly using traditional network addresses for his source and destination addresses. Specifically, Kingdon is referring to a NCP packet that is a part of the NetWare Core Protocol (NCP) (Kingdon, column 2, lines 38-49). It is well known that NetWare does not bind gate names to transport references and further does not include messages gates configured to receive messages at an address including a combination of a gate name and a transport references. Kingdon is clearly describing traditional network addresses that have nothing to do with either binding gate names to transport references or receiving messages at an address including a combination of a gate name and a transport reference.

**Fifth Ground of Rejection:**

Claims 5, 13 and 21 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Marcos in view of Bergman et al. (U.S. Patent 6,564,263 – hereinafter Bergman).

**Claims 5, 13 and 21:**

Claims 5, 13 and 21 are patentable for at least the reasons given above in regard to their respective independent claims.

Furthermore, Marcos in view of Bergman fails to teach or suggest a system wherein each message gate is configured to verify messages according to a data representation language message schema, wherein the data representation message schema comprises an eXtensible Markup Language (XML) schema. In regard to claim 5, the Examiner states, “[w]hile Marcos did not specifically mention the use of XML, it would have been obvious to one of ordinary skill in the art to consider the use of such a format, as Bergman disclosed XML to be both portable, independent of operating environment, and advantageous for linking different modalities of content.” However, as stated above, Marcos teaches using a mediating component that “identifies the expected method specification and arguments for the server” (Marcos, column 6, lines 62-63) and “performs any necessary message translation” (Marcos, column 4, lines 18-23). Also, Marcos uses a proxy object that “determines the expected method identification and the number and type of arguments for the server object” (Marcos, column 4, lines 23-25). Under Marcos, the mediating component “carries out dynamic translation at run-time” (Marcos, column 7, lines 22-23). Thus, it would make no sense in Marcos to verify messages according to a data representation language message schema.

Marcos teaches the translation of method invocations and arguments, hence code, between two object models. However, Marcos fails to teach wherein each message gate is configured to *verify* messages according to a data representation language *message schema*, and wherein the data representation language schema comprises an eXtensible Markup Language (XML) schema.

Bergman discloses XML as “a tagged markup language for representing hierarchical, structured data” (Bergman, column 14, lines 12-14) and that “XML is useful for specifying and maintaining relationships between different modalities and versions, etc. of content” (Bergman, column 14, 20-22) (emphasis added). **Thus, Bergman**

**supports Appellants arguments above, in regard to the various § 102 rejections, that the prior art teaches data representation languages, such as XML, to be used for representing data content, not for programmatic messaging.** Appellants argue that it would be counterintuitive to use XML to represent code such as the method invocations disclosed in Marcos. In the prior art, such as Bergman, XML is used to represent hierarchical, structured data (i.e. content), not to translate code, such as the translated method invocations in Marcos.

In response to Appellants' arguments regarding claim 5, the Examiner argues that Bergman discloses the use of XML for describing method specifications and function arguments and specifically states, "Bergman disclosed it [is] necessary to translate content data in some cases, such translation performed by a proxy" and cites column 5, lines 44-63 of Bergman. The Examiner's interpretation of Bergman is clearly incorrect. The cited reference discusses only the translation of *content* data from a "content server system" in order for the same *content* to be "displayed on different platforms" and that this translation is "necessary since the display and processing capabilities of the various client devices ... may differ widely." Thus, this passage does not mention anything about using XML, or any other data representation language, for describing method specification or function arguments, as the Examiner contends.

The Examiner further argues, "a data representation encapsulating the necessary arguments and methods to perform the translation were described in a description scheme object" and cites column 6, lines 39-67 of Bergman. However, Bergman is referring to a description scheme (InfoPyramid) that describes various data and content modalities. Nowhere does Bergman refer to any sort of representation encapsulating arguments and methods or perform a translation, as the Examiner contends. The cited passage repeatedly refers to the "data model", and "multimedia content." While Bergman discloses that an InfoPyramid defines methods and/or criteria for generating, manipulating, transcoding, and otherwise transforming the source multimedia content, such transformations of multimedia content has no relation to verifying messages

according to a data representation language message schema, and Bergman gives no suggestion that the multimedia content is verified according to any schema, let alone a data representation language message schema.

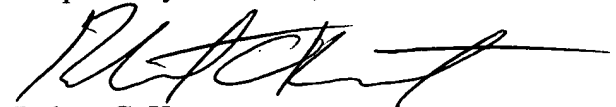
Thus, any combination of Marcos and Bergman would still fail to teach wherein each message gate is configured to verify messages according to a data representation language message schema, and wherein the data representation language schema comprises an eXtensible Markup Language (XML) schema.

### **VIII. CONCLUSION**

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-24 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of \$500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-64200/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,



Robert C. Kowert  
Reg. No. 39,255  
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.  
P.O. Box 398  
Austin, TX 78767-0398  
(512) 853-8850

Date: December 10, 2004



## **IX. CLAIMS APPENDIX**

The claims on appeal are as follows.

1. A device in a distributed computing environment, comprising:

one or more clients;

a plurality of message gates, wherein each message gate is configured for sending and receiving messages for one of said clients in a data representation language to and from a respective paired message gate at another device in the distributed computing environment;

a message transport configured to implement a transport protocol for sending and receiving messages;

wherein each one of said message gates references said message transport, wherein each message gate is configured to send and receive messages independently of the other ones of said message gates while sharing said message transport for implementing the transport protocol for sending and receiving its messages.

2. The device as recited in claim 1, wherein each message gate is configured to verify messages according to a data representation language message schema.

3. The device as recited in claim 2, wherein each message gate is configured to verify type correctness of messages sent or received through that message gate according to the data representation language message schema.

4. The device as recited in claim 2, wherein each message gate is configured to verify format correctness of messages sent or received through that message gate

according to the data representation language message schema.

5. The device as recited in claim 2, wherein said data representation language schema comprises an eXtensible Markup Language (XML) schema.

6. The device as recited in claim 1, wherein each message gate is configured to include an authentication credential with each message sent from that message gate.

7. The device as recited in claim 1, wherein each message gate is configured to verify an authentication credential included with each message received by that message gate.

8. The device as recited in claim 1, wherein each one of the message gates is configured for creating a bi-directional message channel with its said respective paired message gate for accessing a service in the distributed computing environment.

9. A method for sending and receiving messages in a distributed computing environment, comprising:

a first message gate referencing a message transport to send first messages to a first destination address, wherein said first messages are formatted in a data representation language;

a second message gate referencing said message transport to send second messages to a second destination address, wherein said second messages are formatted in a data representation language; and

said message transport implementing a transport protocol for sending said first and second messages to said first and second destination addresses respectively;

wherein said first and second message gates independently share said message transport for sending messages to said first and second destination addresses respectively.

10. The method as recited in claim 9, further comprising said first and second message gates verifying said first and second messages respectively according respective representation language message schemas.

11. The method as recited in claim 10, wherein said verifying said first and second messages respectively according respective representation language message schemas comprises verifying type correctness of the first and second messages respectively according to the respective representation language message schema.

12. The method as recited in claim 10, wherein said verifying said first and second messages respectively according a respective representation language message schema comprises verifying format correctness of the first and second messages respectively according to the respective representation language message schema.

13. The method as recited in claim 10, wherein said representation language schemas comprise eXtensible Markup Language schemas.

14. The method as recited in claim 9, further comprising said first and second message gates each including a respective authentication credential with each message sent.

15. The method as recited in claim 9, further comprising said first message gate creating a first bi-directional message channel with a message gate at said first destination address and said second message gate creating a second bi-directional message channel with a message gate at said second destination address, wherein said

message transport is shared for said first and second bi-directional message channels.

16. The method as recited in claim 9, further comprising:

binding a first gate name for said first message gate to a transport reference for said message transport, wherein said first message gate is configured to receive messages at an address including a combination of said first gate name and the transport reference; and

binding a second gate name for said second message gate to the transport reference for said message transport, wherein said second message gate is configured to receive messages at an address including a combination of said second gate name and the transport reference.

17. A carrier medium comprising program instructions, wherein the program instructions are executable to implement:

a first message gate referencing a message transport to send first messages to a first destination address, wherein said first messages are formatted in a data representation language;

a second message gate referencing said message transport to send second messages to a second destination address, wherein said second messages are formatted in a data representation language; and

said message transport implementing a transport protocol for sending said first and second messages to said first and second destination addresses respectively;

wherein said first and second message gates independently share said message transport for sending messages to said first and second destination addresses respectively.

18. The carrier medium as recited in claim 17, wherein the program instructions are further executable to implement said first and second message gates verifying said first and second messages respectively according respective representation language message schemas.

19. The carrier medium as recited in claim 18, wherein said verifying said first and second messages respectively according respective representation language message schemas comprises verifying type correctness of the first and second messages respectively according to the respective representation language message schema.

20. The carrier medium as recited in claim 18, wherein said verifying said first and second messages respectively according a respective representation language message schema comprises verifying format correctness of the first and second messages respectively according to the respective representation language message schema.

21. The carrier medium as recited in claim 18, wherein said representation language schemas comprise eXtensible Markup Language schemas.

22. The carrier medium as recited in claim 17, wherein the program instructions are further executable to implement said first and second message gates each including a respective authentication credential with each message sent.

23. The carrier medium as recited in claim 17, wherein the program instructions are further executable to implement said first message gate creating a first bi-directional message channel with a message gate at said first destination address and said second message gate creating a second bi-directional message channel with a message

gate at said second destination address, wherein said message transport is shared for said first and second bi-directional message channels.

24. The carrier medium as recited in claim 17, wherein the program instructions are further executable to implement:

binding a first gate name for said first message gate to a transport reference for said message transport, wherein said first message gate is configured to receive messages at an address including a combination of said first gate name and the transport reference; and

binding a second gate name for said second message gate to the transport reference for said message transport, wherein said second message gate is configured to receive messages at an address including a combination of said second gate name and the transport reference.

**X. EVIDENCE APPENDIX**

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

**XI. RELATED PROCEEDINGS APPENDIX**

There are no related proceedings.